Mobile Meteorological Measurements

Fredrik Persson
Tobias Svahn

Department of Electrical and Information Technology
Lund University

Advisor: Bertil Lindvall

January 9, 2011

Printed in Sweden
E-huset, Lund, 2011

Abstract

The increasing population and urbanization in the world demands research in areas connected to meteorological data such as microclimate studies, environmental impact and exposure [1], and agricultural needs. However, as the current resolution of meteorological data is very poor [2], these research possibilities are constricted. To address this problem and thus increase the resolution, a system with sensor equipped cell phones is proposed, making every phone user a potential measurement station.

This thesis investigates the technical aspects of making mobile measurements. For this purpose an Android application and sensor unit prototypes measuring temperature, humidity, and pressure, has been designed. To evaluate sensor performance, test measurements are performed both in a climate chamber and outdoors.

The results show that the main challenge with mobile sensors consists of creating a design with good measurement performance, while still remaining mobile. A possible design for such a sensor unit is proposed in this thesis, but more studies are needed in order to evaluate performance of a large scale user based sensor equipped system.

Acknowledgments

We would like to thank our examiner Anders Ardö and our supervisor Bertil Lindvall, for their inexhaustible guidance and support.

We thank Jonas Ardö for devising the idea behind this thesis and giving us general support.

We thank Andreas Persson, Meelis Mölder, Thomas Holst, and Marcin Jackowicz-Korczynski for supplying the logger, support on logger and sensors, supplying the temperature/humidity sensor, and support on the climate chamber.

Furthermore, we would like to thank Josef Wajnblom and Lars Hedenstjerna for their support on the PCB-design.

Table of Contents

List of Figures

List of Tables

Abbreviations

ABS	Acrylonitrile Butadiene Styrene
ADC	Analog to Digital Converter
AVR	Microcontroller from Atmel
bps	Bits per second
cc	Clock cycle
EEPROM	Electrically Erasable Programmable Read-Only Memory
GND	Electrical Ground
GUI	Graphical User Interface
GSM	Global System for Mobile communications
I^2C	Inter-Integrated Circuit
IP	Internet Protocol
LSB	Least Siginificant Bit
MAC	Media Access Control
NDK	Native Development Kit
OTP	One Time Programmable
PCB	Printed Circuit Board
PTAT	Proportional To Absolute Temperature
RFCOMM	Radio Frequency Communication
RX	Receive
SCL	Serial Clock Line
SD	Secure Digital
SDA	Serial Data Line
SDK	Software Development Kit
SPP	Serial Port Profile
SU	Sensor Unit
TX	Transmission
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
USI	Universal Serial Interface
Vdd	Supply voltage
XML	Extensible Markup Language

Chapter 1

Introduction

Climate stations are scarce in large parts of the world, making the resolution of global meteorological data poor. Coupled with growing needs for a better understanding of how the climate affects health issues [1], agriculture, urban planning, transport infrastructure, energy consumption, and environmental aspects, more localized meteorological data is needed.

There are three monthly data sets used globally [2]; National Climate Data Center (Boulder Colorado), NASA's Goddard Institute for space studies (New York), and the UK Met office Hadley Centre in Exeter together with the University of East Anglia's Climatic Research Unit in Norwich, UK. These three data sets only have a spatial resolution of hundreds of kilometers [2] making local climate studies virtually impossible. There are projects under way to increase the resolution, but these are based mostly in Europe and the United States, leaving large parts of the world unaffected.

1.1 Possible applications

By analyzing data collected from meteorological measurements it can be applied to numerous applications which individuals, companies, organizations, governments etc. can make use of. Some of the possible applications include:

- Agricultural monitoring/prediction [3]
- Real time local weather information/prediction/warning [5]
- Microclimate studies on urban environments [4]
- Individuals environmental impact and exposure [1]
- Movement patterns of citizens
- Health monitoring [6]

Presently, much of the meteorological data is used in the field of environmental studies and as an alert system for potentially dangerous weather conditions. Public service companies use weather information to ensure that the general population has access to power, gas, water, and other neccesities during extreme weather. Emergency services use weather information to both assess future hazards and to help organize ongoing operations, e.g. fire departments who use information on relative humidity to predict the fire risk.

Looking into future applications, farmers could use local weather information to decide on what plants, irrigation, fertilizers, and pesticides to use to optimize the growth of their crops. In the medical field, patients with respiratory illnesses could use meteorological data to avoid areas with heavy air pollution. With increasing populations and urbanization, even more environmental research is needed on e.g. microclimates, which consists of a local zone where the climate differs from its surroundings. These effects are very apperant in large cities such as Mexico City, housing almost 9 million inhabitants. Solar radiation is trapped in man made structures during daytime and then released during night time, leading to an increase in temperature compared to its surroundings.

1.2 Problem specification

Presently, the meteorological data available has a spatial resolution of hundreds of kilometers and an hourly or larger temporal resolution. Also, some parts of the world has no coverage at all, leading to the question:

How could the spatial and temporal resolution of meteorological measurements be increased?

There are three important aspects of meteorological measurements; time, location, and accessability of the measurement data. These properties are already present in cell phones and as there are more than 4 billion registered world wide, one possible solution to increase the resolution of meteorological data is to integrate climate sensors into cell phones. These sensors would not have the same accuracy as the standardized meteorological stations used in the three global data sets, but would together provide a higher spatial resolution and therefore pave way for new possibilities in meteorological measurements.

This thesis aims at investigating the possibilities of connecting a sensor unit to a cell phone for the purpose of sending meteorological data to a server. The core of the problem revolves around constructing a prototype for the sensor unit and a cell phone application which handles communication.

There exist several different communication alternatives between sensor and phone. The most common protocols available on cell phones are USB and Bluetooth. Depending on the protocol chosen, the implementation of the power supply to the sensor unit will differ. A large advantage of wireless communication protocols is the fact that the sensor can be placed freely compared to the cell phone which often resides in the users pocket or hand bag. To associate measurements to geographical locations, either GPS or Cell identification will be used.

A climate chamber test and real world tests will be done to verify the performance of the sensor unit, both the difference in accuracy between the units and the accuracy of the measurements themselves will be measured.

1.3 System design

The overall system architecture consists of three parts; sensor unit (SU), cell phone and a server. On a request from the cell phone, the SU measures temperature, humidity and pressure. The raw data from these readings are then sent to the cell phone where they are processed into human readable format. Information regarding location and time are added to the readings. This information is then either stored to file or sent to a server.

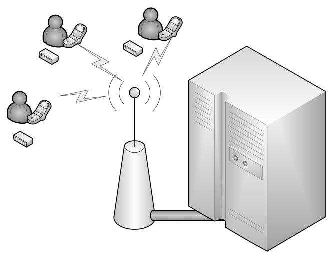


Figure 1.1: System architecture with sensors, users, cell phones, and server.

The SU is designed to work with any software platform through Bluetooth, as long as the correct communication commands are sent and received.

1.4 Thesis organization

Chapter 2 presents the theory used in this thesis. Information is presented regarding communication protocols, cell phone operating systems, sensors, sensor measurements, and a microcontroller.

Thereafter, chapter 3 deals with the components inside the sensor unit itself, e.g. sensors, power regulator, and the Bluetooth module.

Chapter 4 describes the system architecture of the sensor unit and its associated cell phone application in detail.

In chapters 5, 6, 7, and 8, verification, results, discussion, and conclusions are presented respectively.

Chapter 2

Theory

2.1 Protocols

This section provides a brief overview of the different communication protocols used in the sensor unit.

2.1.1 Inter-Integrated Circuit [7][8] (I^2C)

I^2C is a two wire, 8-bit serial communication protocol capable of speeds up to 3.4 Mbit/s in high-speed mode. It is based on a (multi)master-slave relationship with an address space of 7 bits, making it possible to address 112 slaves, as there are 16 reserved bus addresses. As an extension to the standard, it is possible to use 10-bit addressing, which will yield an additional 1024 slave addresses. However, the number of devices connected is limited by the capacitance load on the bus, which may not exceed 400 pF. The general device address consists of a 7 bit address and a single write/read bit.

The I^2C protocol defines two unique conditions, Start and Stop, which are sent by the master. A Start condition is set when clock (SCL) is high and data (SDA) is on a falling edge, while a Stop condition is set when SCL is high but SDA is on a rising edge. After the Start condition, the slave address is put on the bus, with the LSB controlling if it will be a read or write transfer. Thereafter, the slave should acknowledge by pulling SDA low during the 9th cc, see figure 2.1. Depending on the write/read bit, data will either be received from the slave or more commands/addresses will be sent from the master. After each byte the slave will send an ACK.

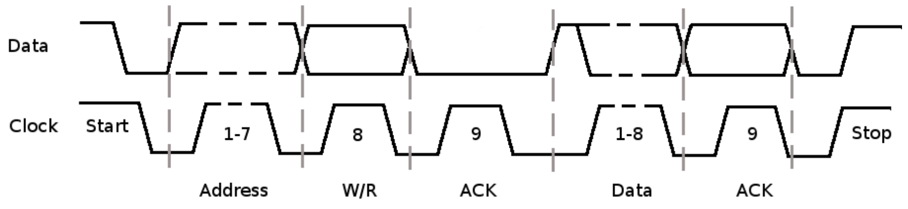


Figure 2.1: I^{2}C wave form during a W/R operation. The bus is considered busy from the Start condition until a while after the Stop condition.

If the slave device needs more time before sending data, it can force the SDA line high, which will make the master wait until the slave releases the SDA line. Clock generation on the bus is handled by the master in the system.

2.1.2 Sensibus [9]

Sensibus is a protocol developed by Sensirion that shares many similarities with I^{2}C. It is an 8-bit, Master-Slave two wire interface consisting of a clock (SCK) and data line (DATA). A start condition is initiated by pulling DATA low while SCK is high and is completed by raising DATA the next time SCK is high. After a start condition, the following 8 bits consists of three address bits and five command bits. After a slave has been addressed, it should send an ACK during one cc after the falling edge of the SCK. The slave will now enter measuring mode, holding DATA high. To indicate when the measurement is completed, DATA will be pulled low. The master will then restart the clock to receive the data and then ACK each byte by pulling DATA low. One byte of the data field consists of an optional CRC checksum. Clock generation on the bus is handled by the master in the system.

2.1.3 Universal Asynchronous Receiver/Transmitter [12] (UART)

UART is a serial, asynchronous communication standard. As it is asynchronous, no clock is shared between the sender and receiver. Instead predefined parameters are used, these include baud rate, packet structure, and flow control.

Data from an UART sender might be sent at a faster rate than the receiver can process, eventually leading to a buffer overflow. Therefore, flow control is used to avoid possible data loss.

A general packet sent over UART contains a start bit, data bits, an optional parity bit, and one or more stop bits.

2.1.4 Bluetooth [13]

Bluetooth is a wireless communication standard employed in the unlicensed 2.40-2.483 GHz band. To enable communication between several units at a time, time-slotted frequency hopping is used.

A Bluetooth network consists of one master, providing the synchronization reference with up to 7 slaves, called a Piconet. To form a larger network, one of the slaves can become the master in a new Piconet, forming a Scatternet. Due to frequency hopping, a maximum of 80 units can communicate simultaneously within a Scatternet.

The Bluetooth standard is built upon layers, the 4 lowest being; Radio, Link Control, Link Manager, and Logical Link Control and Adaptation Protocol, which are all transport protocols. Above these are the middleware protocols such as Radio Frequency Communication (RFCOMM) and Service Discovery Protocol (SDP). At the top layer, applications and profiles can be found. Each device supports at least one of the predefined Bluetooth profiles. A profile describes what parts of the Bluetooth stack and which protocols are implemented.

The Serial Port Profile (SPP) is used to create a virtual serial cable between devices. SPP uses the RFCOMM protocol to establish a connection.

The Bluetooth standard has several parameters that control the behavior of the Bluetooth device. Two important settings are the page scan and the inquiry scan window and intervals. The inquiry scan is related to the discoverability of devices. The interval time sets the time between each window. If a device receives an inquiry during an inquiry scan, it will answer with an inquiry response. The response contains information necessary for the inquiring device to connect to the responding device. With a shorter inquiry scan window, the time during which a Bluetooth device can be discovered is decreased.

The page scan corresponds to the connectability of a device. A device issues a page to a specific Bluetooth device in a page scan window, which then responds. A shorter page scan will lead to a shorter time window where a connection can be made.

2.2 Android [14]

Android is an open source, mobile operating system project led by Google. The core is based on the Linux kernel, which handles basic functionality such as process management and networking. Placed above the kernel are system libraries and the Android runtime environment. The top two layers consist of the Application framework and the Application layer. There is both a SDK and NDK available for developers. Most of the OS runs on Java while certain low level routines rely on C/C++.

Android applications are based on four main components; activities, services, content providers, and broadcast receivers. The activity is the main building block and contains the visual interface. They are most often presented in full screen windows to the user. Services are long running application components, most often placed in the background. The content providers share data between applications. Broadcast receivers are used to receive intents, where an intent contains an action to be performed.

Bluetooth in Android is based upon the open source project BlueZ [28] where the API was introduced in the SDK for Android 2.0.

2.3 Meterological Measurments

2.3.1 Temperature [15]

Temperature is closely related to thermal energy, which describes the kinetic energy contained in matter. However, temperature describes the mean kinetic energy in matter, see equation 2.1.

$$\overline{E}_k = \frac{1}{2}kT \tag{2.1}$$

$$k = \textit{Boltzmann's constant}$$
$$T = \textit{temperature}$$

2.3.2 Humidity [11]

Humidity is a measure of the amount of water vapor available in a gas. There are different ways to represent humidity; absolute, relative, dew-point, mixing ratio, and Heat Index. The most common way to represent humidity in meteorological terms is in relative humidity.

Relative humidity is calculated by the following formula:

$$U(t) = \frac{e_w}{e_w^*(t)} \cdot 100 \tag{2.2}$$

where e_w is the partial vapor pressure and $e_w^*(t)$ is the saturated vapor pressure.

The saturated vapor pressure is calculated by:

$$e_w^*(t) = \alpha e^{\frac{\beta \cdot t}{\lambda + t}} \tag{2.3}$$

$$\alpha = 6.112hPa$$
$$\beta = 17.62$$
$$\lambda = 243.12^\circ C$$

Because of the relation given by equations 2.2 and 2.3, it is important that the temperature and humidity measurements are done in close proximity for accurate results.

2.3.3 Pressure [16]

Pressure is the orthogonal force measured on an objects surface. The SI unit for pressure is Pascal (Pa), which is defined as N/m^2. The atmospheric pressure is generated by the amount of air per unit area above the surface of an object. The mean atmospheric pressure at sea level is 101325 Pa. Equation 2.4 describes how pressure is calculated.

$$P = \frac{F}{A} \tag{2.4}$$

$$\begin{aligned} F &= Force \\ A &= Area \end{aligned}$$

As the atmospheric pressure varies with the altitude, the height above sea level can be calculated by using the pressure at sea level in combination with the measured pressure. See equation 2.5 [8].

$$Altitude = 44330 \cdot (1 - (\frac{p}{p_0})^{1/5.255}) \tag{2.5}$$

$$\begin{aligned} p &= measured\ pressure \\ p_0 &= pressure\ at\ sea\ level \end{aligned}$$

Chapter 3

Sensor unit components

The sensor unit consists of a microcontroller (AVR), Bluetooth module, digital pressure sensor and a digital temperature/humidity sensor, all mounted on a PCB and placed inside a box. Three NiMH batteries, each rated at 1.2V connected in series, provide power to the SU.

The Bluetooth module is connected to the AVR through UART. The two sensors are also connected to the AVR. They share clock lines but use separate data lines. The pressure sensor uses the I^2C protocol, whereas the temperature/humidity sensor uses Sensibus.

As the microcontroller lacks hardware support for any of the protocols used in the SU they were implemented in software instead. This solution works well for low speed and low cost applications, but requires more work than using microcontrollers with specific hardware support. To facilitate the implementation of UART on the microcontroller, the USI module is used for I/O buffering and timing purposes. All data passed from the AVR to the Bluetooth module is forwarded to the currently connected Bluetooth device. Figure 3.1 shows an overview of the system.

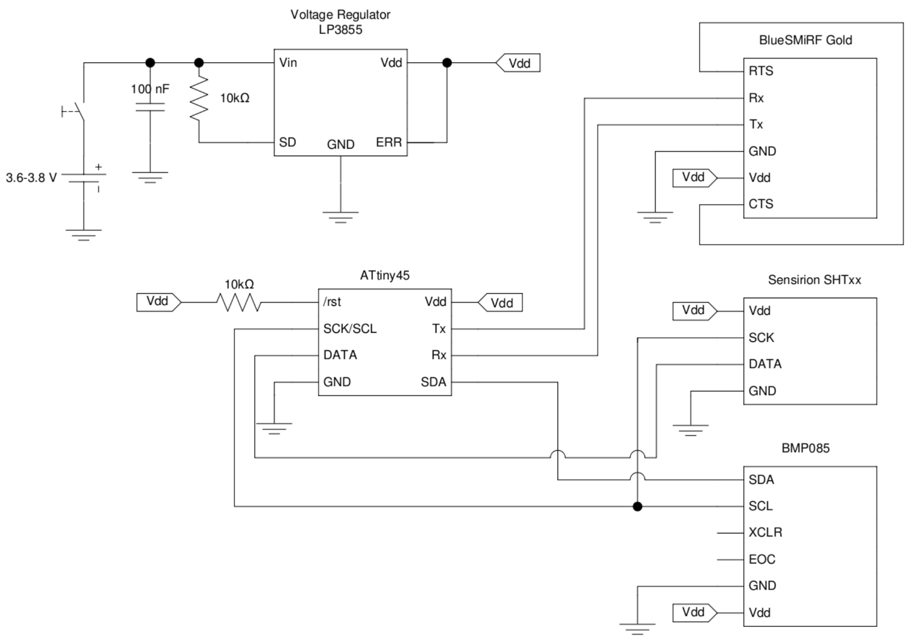


Figure 3.1: Schematic overview of the sensor unit.

3.1 Atmel ATtiny45 [17]

The Atmel AVR microcontroller family consists of many different models, ranging from 8 to 32 bit controllers. The ATtiny45 model is an 8-bit, low-power RISC microcontroller. It includes 4kB programmable flash, 256B SRAM, and 6 I/O lines. Additionally, it has hardware support for ADC, USI, counters, and interrupts. This model fits the requirements of low power, small size, and good development tools. It is also well documented and has a large user base.

The microcontroller can be configured to operate in different modes, clock speeds, etc. This is done by writing to specific registers called fuses.

The package of the microcontroller consists of 8 pins; Vdd, GND, and PB0-PB5. Table 3.1 shows the pin assignments used in this design.

Pin name	Usage
PB0	I^2C Data
PB1	RX
PB2	TX
PB3	Clock
PB4	Sensibus Data
PB5	RESET (active low)

Table 3.1: AVR pin assignment

3.2 Voltage regulator [18] (LP3855)

The SU operates in an interval of 3.3 - 3.6V, where the pressure sensor sets the upper limit and the Bluetooth module sets the lower limit. As the power source consists of batteries and the average current load from the SU is low, each battery will provide more than the specified 1.2V when fully charged. Because of this a voltage regulator is used to fix the supply voltage to 3.3V. Important to note is to choose a voltage regulator with a low dropout voltage, as a high dropout will reduce the battery time.

3.3 Bluetooth module

The Bluetooth module (BlueSMiRF Gold) used in the SU is manufactured by SparkFun Electronics. It was chosen as it is well documented and includes an internal voltage regulator with an input span of 3.3 - 6V and an output of 3.3V, facilitating circuit design.

It supports SPP and DUN[1] profiles and acts as a RX/TX pipe, which is possible due to the built in Bluetooth stack[2]. It is a class 1 module with an on-chip antenna [20].

The module has 6 pins; Vdd, GND, TX, RX, CTS (Clear To Send), and RTS (Ready To Send).[21].

Usually, the RTS and CTS pins are used for hardware flow control over the UART, ensuring that the receiver is ready for incoming data. However, as data is only sent in short bursts with a low baud rate in this design, the RTS and CTS pins are not needed.

3.4 Temperature/Humidity Sensor [9][10] (SHTxx)

The Sensirion SHTxx sensor houses a capacitive sensor element for humidity measurements and a band-gap PTAT element for temperature measurements. Furthermore, the sensor contains an amplifier, ADC, OTP memory and a control unit. There are several advantages with these sensors; the sensor elements for humidity and temperature measurements are placed close in proximity for good performance, a fully digital interface eliminating the need of external ADC, and low power consumption with automatic sleep mode. The internal logic of the sensor combined with the custom communication protocol (Sensibus) enables easy operation.

The analog values from the sensor elements are converted by the internal ADC. The control unit interfaces with the bus connection, OTP and the sensor elements through the ADC. The OTP memory is used to store calibration coefficients. As the sensors are factory calibrated, each sensor will have unique calibration coefficients that is automatically applied to the sensor output by the sensor logic.

[1]Dial-up Networking

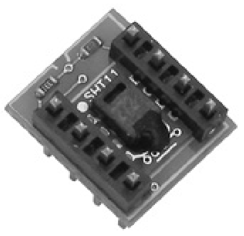
[2]Protocols supported: GAP, SDP, RFCOMM, L2CAP

Sensor output is in raw format and needs to be converted, see appendix A for the required equations. Technical specifications for the sensor are presented in table 3.2.

Parameter	**Min.**	**Typ.**	**Max.**
Operating range	-40°C		123.8°C
RH accuracy @ 25°C		±3%	±5%
Temp. accuracy @ 25°C		±0.4°C	±2.6°C
Power comsumption	2μW		3 mW

Table 3.2: Specifications for the temperature/humidity sensor.

Two different sensor packages have been used for the temperature/humidity sensor; the SHT11 with breakout board from Parallax [19], and the SHT71 [10], see figure 3.2. The break out board from Parallax facilitates circuit design and construction by incorporating premounted pull-up resistors and pins for easy soldering. The SHT71 was also tested as it was suspected to produce faster response times as the package design should allow for better air flow around the sensor.



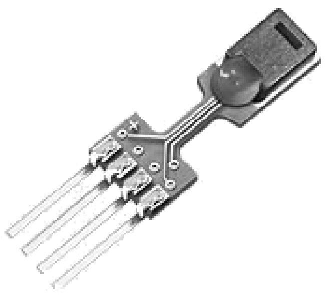


Figure 3.2: Left: SHT11 package mounted on breakout board. Right: SHT71 package.

3.5 Pressure Sensor [8] (BMP085)

The Bosch BMP085 is a piezo resistive pressure sensor which consists of four main parts; sensor elements, ADC, control unit, and an EEPROM. Advantages include; fully digital interface, built in temperature compensation for the piezo resistive elements, and it uses a standardized communication protocol (I^2C) that allows for multiple units on a single bus.

The ADC handles conversion of measurement values from the sensor element into digital values. The control unit interfaces with the bus connection, EEPROM and the sensor element through the ADC. As sensors will have different characteristics due to production environment, each sensor is calibrated. These calibration values are stored in the EEPROM, but in contrast to the SHTxx sensor, need to be applied to the sensor output manually by the user.

Additionally, the sensor measures temperature and uses these readings to compensate the pressure readings. This is because of the temperature dependency in the piezo resistive elements.

To switch between temperature and pressure mode, a value is written to a control register on the sensor.

See table 3.3 for technical specifications on the sensor.

Parameter	**Min.**	**Typ.**	**Max.**
Operating range	-40°C		85°C
Operating range (full accuracy)	0°C		65°C
Pressure accuracy @ 0 - 65°C		$\pm 1 hPa$	
Pressure accuracy @ -20 - 0°C		$\pm 1.5 hPa$	
Temp. accuracy @ 0 - 65°C		±1°C	
Current consumption	0.1μA		650μA

Table 3.3: Specifications of the pressure sensor

As the sensor output is in raw format it needs to be converted, see appendix B for the required equations.

3.6 PCB-layout

To minimize heat transferred from the AVR, voltage regulator, and Bluetooth module to the SHTxx temperature/humidity sensor, the PCB is divided into two parts, a bottom plate and a riser. Also, to keep the production complexity to a minimum, the PCB routing has been done on a single layer.

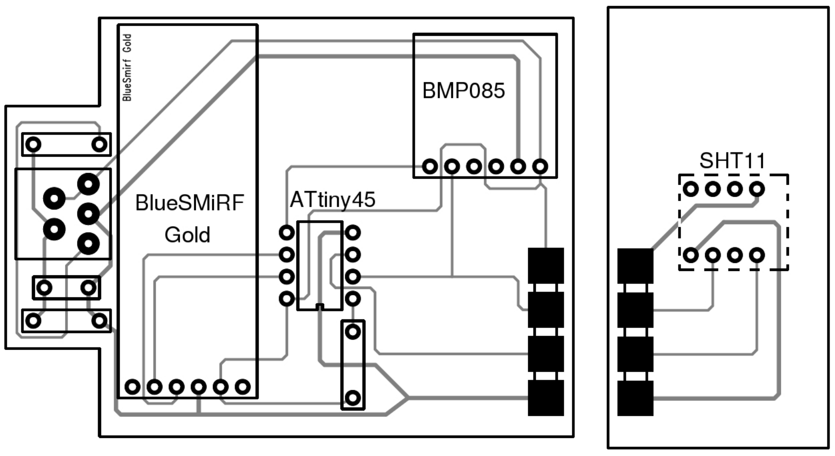


Figure 3.3: Printed circuit board. Left: Bottom plate. Right: Riser (Not actual size).

The bottom plate consists of a power regulator, Bluetooth module, AVR, and a pressure sensor. All components, except the pressure sensor, are placed as far away from the temperature/humidity sensor as possible. The riser only holds the temperature/humidity sensor. As the riser is mounted orthogonally to the bottom plate, a barrier is created separating the sensor from the other components. The riser is soldered and glued to the bottom plate.

3.7 Box design

The SU box is made of ABS-plastic with the dimensions 85x40x56mm (LxHxW). The requirements of the box included sufficient space for components and battery pack. To provide ventilation, holes were drilled in the box, see figure 3.4. The battery holder is mounted in the lid of the box. The riser of the PCB is adjusted to fit into the rails in the box, providing stability. The riser also creates two compartments in the box, see figure 3.5. The smaller compartment houses only the temperature/humidity sensor, while the larger compartment contains the rest of the components. In front of the temperature/humidity sensor there is a radiation shield, in the form of an aluminum sheet. On the opposite side of the radiation shield there is an on/off switch which controls power from the battery.

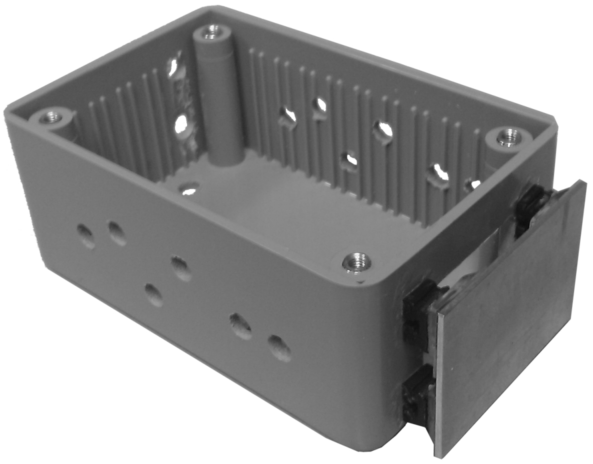


Figure 3.4: Sensor unit box.

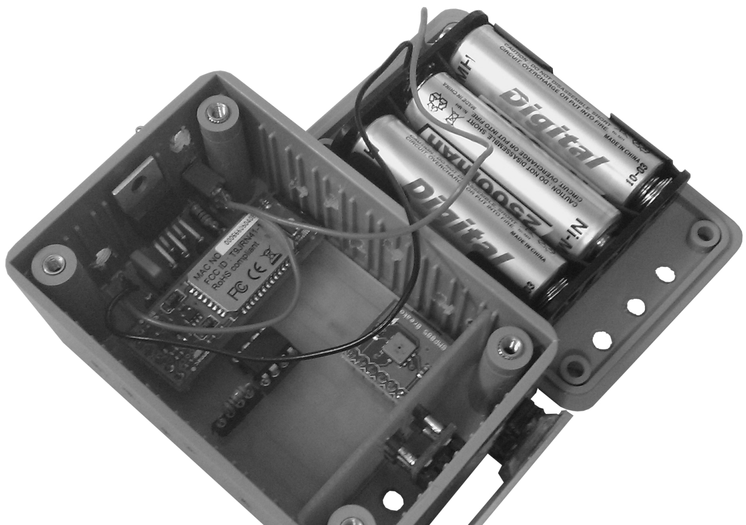


Figure 3.5: Complete sensor unit.

Chapter 4

System architecture

In this implementation, Android was chosen for the cell phone application since it is a rapidly growing operating system available on a large range of devices, e.g. cell phones, tablets, and TV's. The operating systems Maemo [23], iOS [22] and Symbian [25] were also considered, but Maemo and iOS are only available on a few specific high-end cell phones. Symbian on the other hand has a very large user base but is very fragmented [24], i.e. it is available in many different versions making united application development difficult.

4.1 Sensor Unit

4.1.1 Pressure Sensor (BMP085)

A measurement with the pressure sensor has the following procedure:

1. Retrieve the calibration coefficients. This is only done if the calibration coefficients for the specific sensor haven't been fetched before.
2. Set the control register to measure temperature.
3. Retrieve the raw temperature data.
4. Set the control register to measure pressure.
5. Retrieve the raw pressure data.

   See figure 4.1 for a complete system implementation.

The AVR obtains data from the sensor (1, 3, 5) by generating the I^{2}C start condition and then addressing the sensor with the write bit set. After receiving an ACK, the AVR will send the address of the desired data. To read the value, a repeated start condition is sent, followed by the sensor address with the read bit set. The sensor will then put the requested data on the bus. When the data has been received in the AVR, it will generate a NACK and the stop condition.

To set the control register (2, 4), the AVR generates the start condition and addresses the sensor with the write bit set. After receiving an ACK, the AVR will put the address of the control register on the bus and wait for an ACK. To put the sensor in the desired mode, the corresponding control register value is put on the bus.

4.1.2 Temperature/Humidity Sensor (SHTxx)

To initiate a temperature measurement, the AVR generates the Sensibus start condition and puts the sensor address combined with the command bits on the bus. After receiving an ACK, the AVR waits until the measurement on the sensor is finished, which is indicated by the sensor pulling the data line low. The AVR then restarts the clock to retrieve the data from the sensor. Each received byte is followed by an ACK from the AVR.

In contrast to the pressure sensor, the calibration coefficients are automatically applied to the raw sensor output by the internal sensor logic.

See figure 4.1 for a complete system implementation.

4.1.3 Bluetooth Module (BlueSMiRF Gold)

The Bluetooth module needs to be configured to operate with both the Bluetooth master and the AVR. Table 4.1 specifies the essential settings. As each SU uses the MAC-address as a unique identifier, each Bluetooth module is also given the name MMMSensor-XXXX, where the X's represent the last four bytes of the MAC-address.

Settings	**Value**
Profile	SPP
Mode of operation	Slave
UART data mode	8 data bits, parity None, 1 stop bit
Baud rate	2400 bps
Name	MMMSensor-XXXX
Authentication key	1234
Page scan window	1% duty cycle
Inquiry scan window	1% duty cycle
Transmission strength	-20 dBm

Table 4.1: Bluetooth module configuration. See reference [26] for command set.

4.1.4 AVR (ATtiny45)

The AVR receives commands from the Bluetooth module. A total of six commands can interpreted, see table 4.2. Each interpreted command is answered by a NACK or ACK, depending on if the command is recognized or not. The frequency is set to 1 MHz.

Function	**Command value**	**Number of bytes**
Get SHTxx data	49	4
Get BMP085 temperature	50	2
Get BMP085 calibration values	51	22
Get BMP085 pressure	52	2
Multiple measurements	53	40
Sensor unit information	54	3

Table 4.2: AVR functions and command values

The first four functions retrieve corresponding data from respective sensor. The multiple measurements function takes five consecutive measurements from all the sensors and sends the data back to the Bluetooth module. The last function returns firmware version and the model/version number for the temperature/humidity sensor. Depending on which version of the sensor is returned, the Android application will use the corresponding constants in table A.1.

All communication on the SU is controlled by the AVR. It is both the I^2C and Sensibus master as well as one end of the UART bus. Figure 4.1 describes the complete system implementation.

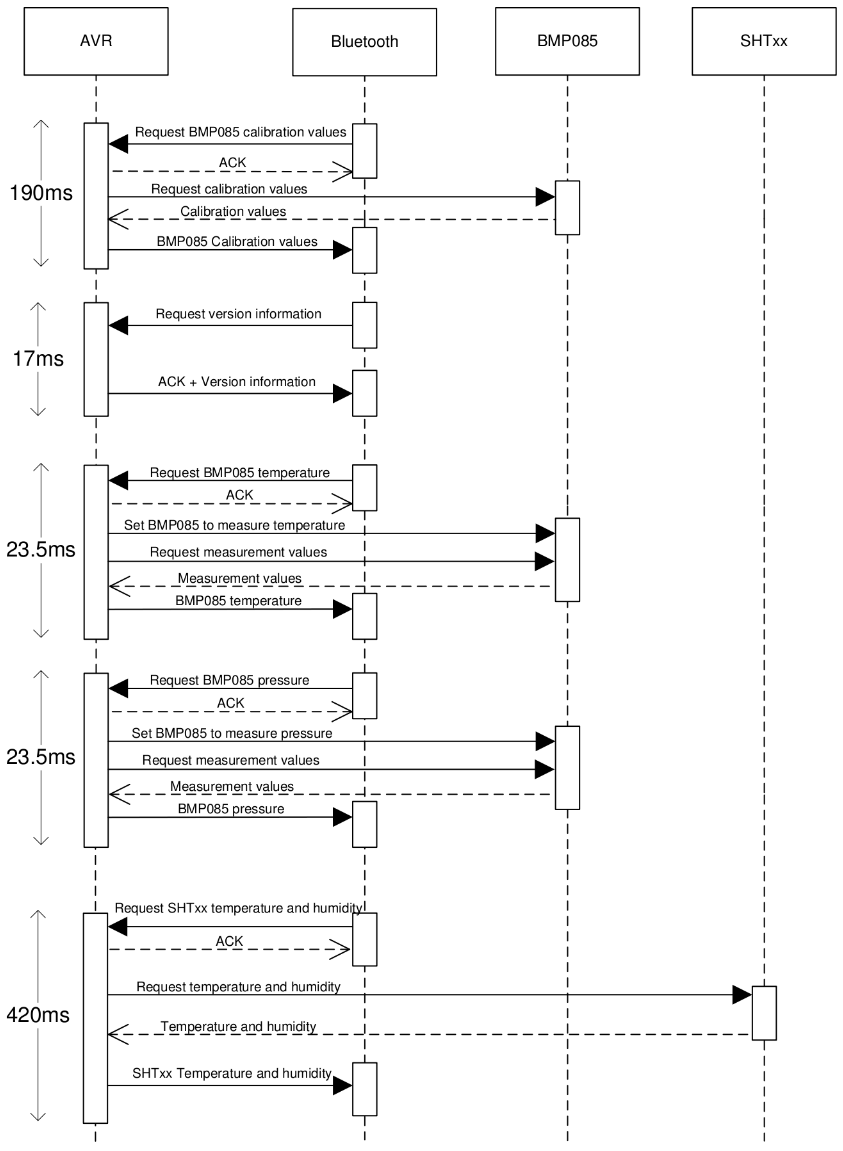

AVR
Bluetooth
BMP085
SHTxx
190ms
Request BMP085 calibration values
ACK
Request calibration values
Calibration values
BMP085 Calibration values
17ms
Request version information
ACK + Version information
23.5ms
Request BMP085 temperature
ACK
Set BMP085 to measure temperature
Request measurement values
Measurement values
BMP085 temperature
23.5ms
Request BMP085 pressure
ACK
Set BMP085 to measure pressure
Request measurement values
Measurement values
BMP085 pressure
420ms
Request SHTxx temperature and humidity
ACK
Request temperature and humidity
Temperature and humidity
SHTxx Temperature and humidity

Figure 4.1: Internal communication in sensor unit. As this is an initial measurement on the SU, the calibration values from the pressure sensor have to be retrieved. Times stated represent worst case.

4.2 Android Application

The Android application consists of four main parts: GUI, data collection, data storage and data transmission. Data is collected both from the SU and from internal sensors. To keep the complexity of the application to a minimum it has been designed with a minimum of activities.

To keep the GUI responsive, demanding tasks have been threaded. The main thread contains the GUI and the object instantiation. It also contains the Timer-Task object, which handles the time scheduling of the automated and custom measurements. Data collection is done in two separate threads; internally (cell phone) and externally (SU). Finally, the data transmission is also handled in a separate thread.

Figure 4.2 demonstrates the events during an initial measurement, with the data later sent to a server.

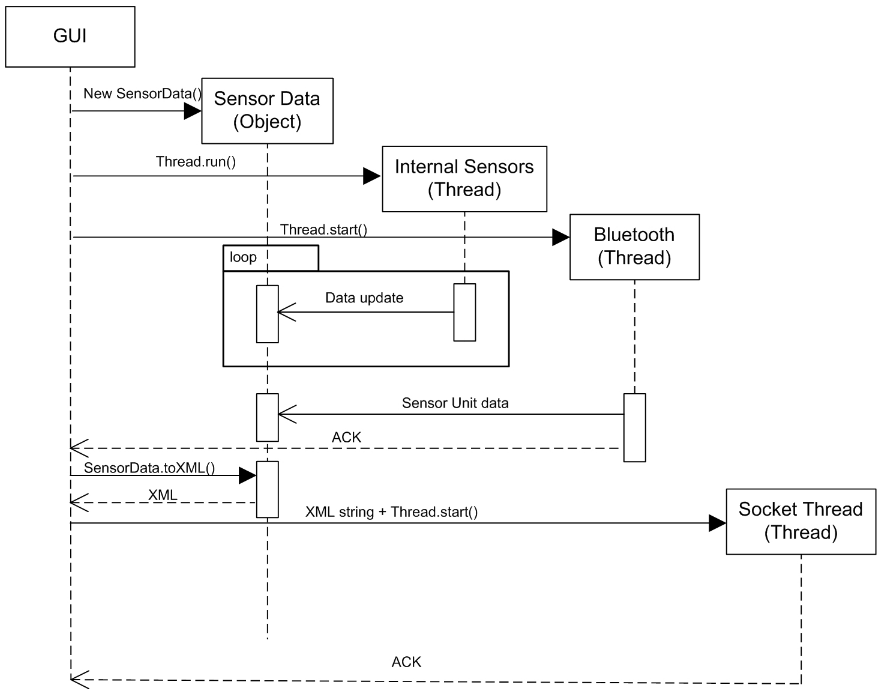


Figure 4.2: Initial measurement as seen from the Android application.

4.2.1 GUI

The GUI consists of four main menus, which are accessible through four buttons on the bottom of the screen.

The Standard Series menu contains a comment field and a start/stop button, which toggles a standardized measurement series. Once started, measurements will be taken at a predefined time interval.

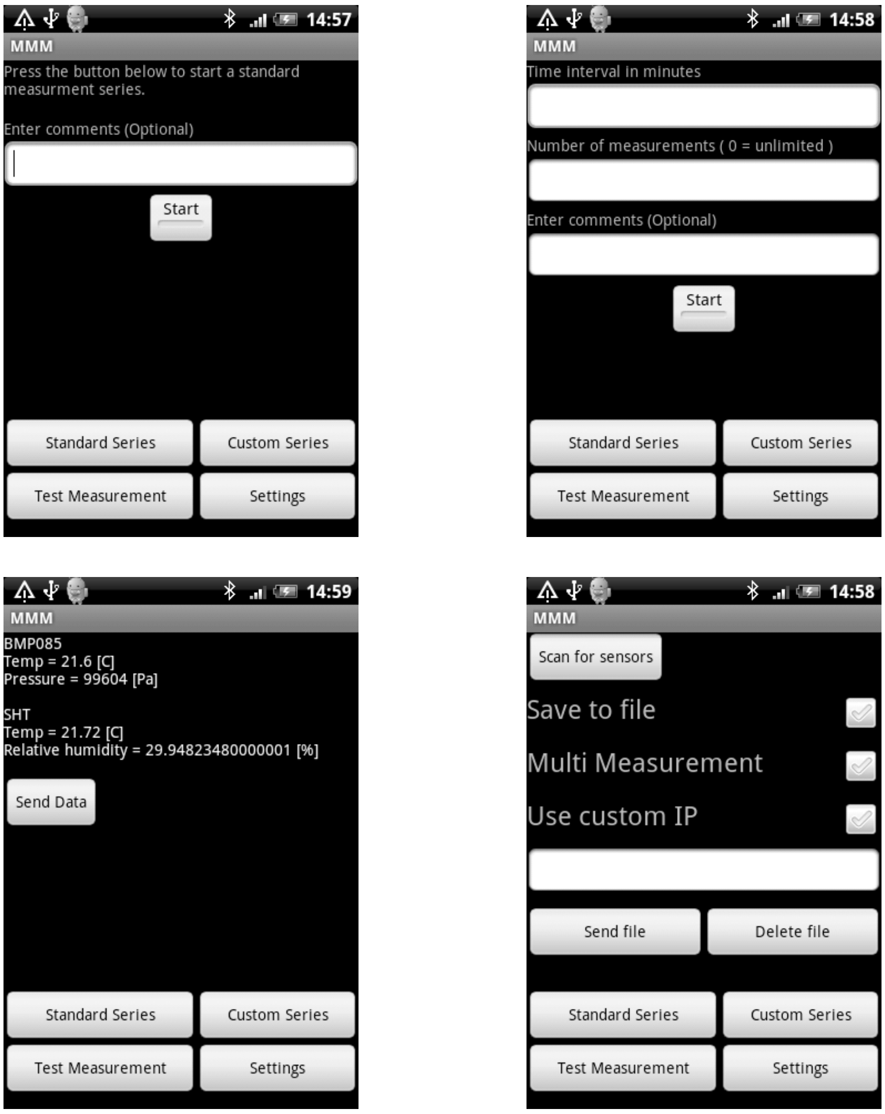
The Custom Series menu consists of a time interval, number of measurements, and a comment field. This enables the time interval and number of measurements to be customizable.

The Test Measurement takes a single measurement from the SU and displays the results. These results can either be sent to the server or simply discarded.

The Settings menu contains the settings available to the user. To see what SUs are available, there is a scan button which will display any units in range. There are three check boxes available. The first controls if the data should be saved to file instead of being sent to a server. Below this check box, there is a Send and a Delete file button.

The second check box toggles if multiple measurements or a single measurement should be done on the SU during each measurement cycle.

The last check box toggles if a custom server IP address is used or not, which can be specified in a text field.



MMM

Press the button below to start a standard measurment series.

Enter comments (Optional)

MMM

Time interval in minutes

Number of measurements (0 = unlimited)

Enter comments (Optional)

MMM

BMP085
Temp = 21.6 [C]
Pressure = 99604 [Pa]

SHT
Temp = 21.72 [C]
Relative humidity = 29.94823480000001 [%]

MMM

Figure 4.3: GUI in Android application. Top left: Standard Series. Top right: Custom Series. Bottom left: Test Measurement. Bottom right: Settings.

4.2.2 Data collection

The external data collection thread collects data through a Bluetooth connection. This thread sends request commands[1] to the SU, which will return measurement data. After the raw data has been received it will be converted into its corresponding units. The required equations for these conversions can be found in appendix A and B.

The internal data collection thread registers listeners for the internal sensors. Internal sensors used are; GPS, accelerometer, and the light sensor.

4.2.3 Data storage

Data storage is a two step process. The data is first stored temporarily in a synchronized object, containing relevant data variables. If file storage is enabled in the application, retrieved data from the SU and internal sensors will be stored to the SD card. One saved measurement on the SD-card roughly corresponds to 800 bytes depending on the amount of comments and what sensors are included in the measurement.

When the data is retrieved from the data storage object, either to be saved to file or be sent to the server, it is converted to a predefined XML-format. During the XML conversion, the latest GPS fix is checked. If no GPS fix has been acquired during the last 10 seconds, cell identification or Wi-Fi location will be used instead for positioning. See appendix C for the predefined XML-format.

[1]See table 4.2 for available commands.

Chapter 5

Verification

To verify the performance of the sensors, a controllable environment and a reference is needed. To test the consistency of the sensors, three SUs have been constructed and then tested in a climate chamber with the possibility to control temperature and humidity. Tests were also conducted outdoors to test real world performance. The reference used in the measurements was the Campbell Scientific CS215 [27] temperature and humidity probe and VAISALA PTB110 pressure sensor, both connected to a data logger. These sensors were supplied by Geocentrum, LTH.

Two data series were programmed in the climate chamber. In series1, the temperature was the controlled parameter. It was programmed to change from low (4°C) to high (40°C) and low (4°C) again, with as high ΔT as possible, while the humidity was programmed to a constant value of 50% RH. In series2, humidity was the controlled parameter. The humidity was set to represent a range between 30 - 85% RH with a constant temperature of 25°C.

During testing in the climate chamber it was dicovered that the relative humidity could not be controlled. Instead the changes in humidity are due to the temperature changes.

To test real world performance, two measurement series according to the following pattern were recorded:

- Start on the 4th floor of the E-building, LTH Lund.
- Take the elevator down to the 1st floor.
- Take a walk according to the plotted routes in figure 6.6 (walk1) and 6.13 (walk2).
- Return to the 4th floor by elevator.

During walk1, the measurement equipment was placed in a bicycle basket. In walk2, the equipment was attached to a backpack.

In the real world performance tests, the cell phone application was used as data logger, while a laptop was used during the climate chamber measurements.

To verify the pressure measurements, the results were compared with the VAISALA reference sensor during atmospheric pressure changes. Furthermore, to ensure that the pressure sensors were not affected by temperature change, equation 2.5 was used to calculate the height difference during walk1. This data was then later compared to the height elevation of the same route plotted in Google Earth.

Three SHT11 sensors were ordered for the SUs. Of the delivered sensors, two were V4 and one was V3. Therefore during the measurements in the climate chamber, SU 1 and SU 2 used a SHT11 V4 sensor while SU 3 was equipped with a SHT11 V3 sensor. As the results from the heat inertia tests were not entirely satisfactory, the SHT11 in SU 2 was replaced with a SHT71 V3, as it was suspected that the SHT71 had a better response time due to its packaging. Therefore, all other measurements were done with SU 2 using SHT71.

To measure the power consumption a 10Ω resistor was connected in series with the power switch. The voltage drop was then measured to obtain the current drawn by the circuit. The voltage was measured both when the Bluetooth module was connected and when it was idle.

To verify that the Android application worked properly it was tested on four different phone models; HTC Desire (Android 2.2), HTC Legend (Android 2.1/Android 2.2), HTC Hero (Android 2.1/2.2), and Sony-Ericsson x10 mini-pro (Android 2.1).

The measurement setup used outdoors can be seen in figure 5.1.

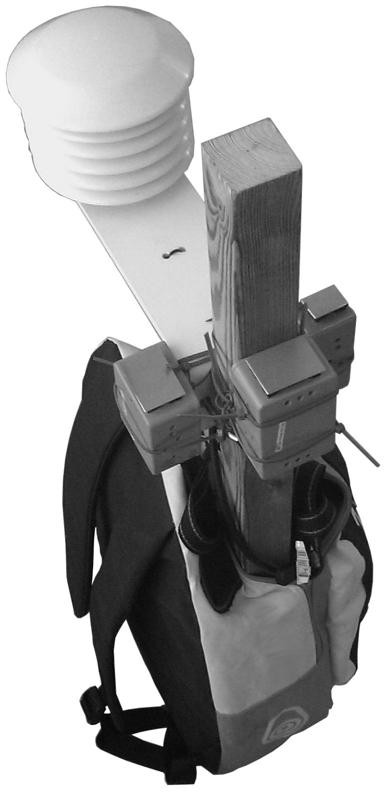


Figure 5.1: Measurement setup used outdoors consisting of the CS215 probe (Not visible due to radiation shield), logger, and the three sensor units.

Chapter 6

Results

The figures in this chapter illustrate the results from measurements with the three SUs compared to the reference during various tests. Further analysis of these results is done in chapter 7.

6.1 Climate chamber measurements

6.1.1 Series1

This section presents temperature, humidity, and pressure measurements from the climate chamber, series1. The results in this section are the mean of every five measurements, taken once a minute.

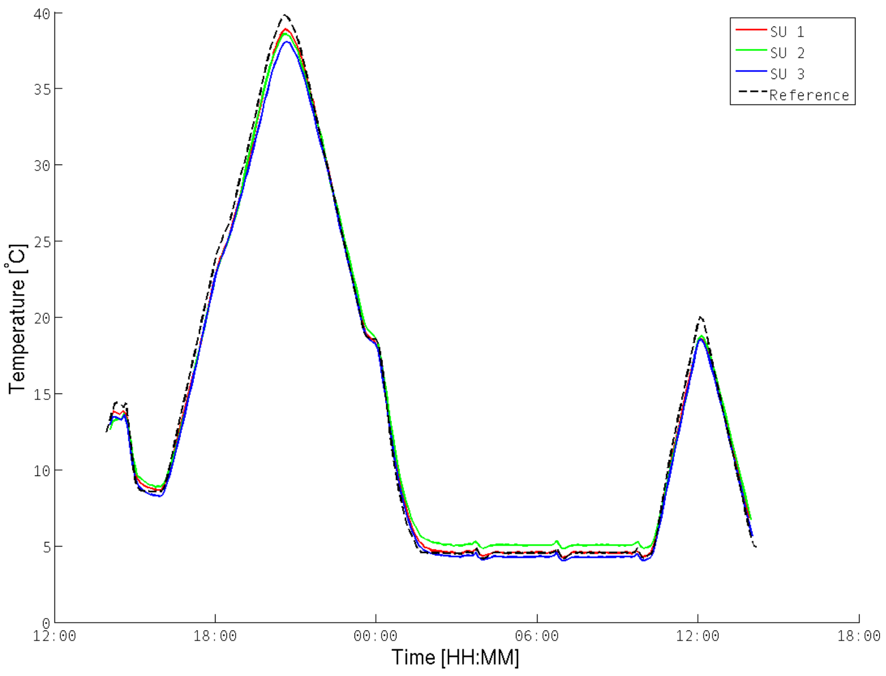


Figure 6.1: Temperature measurements in climate chamber, series1. SUs and reference produce very similar results.

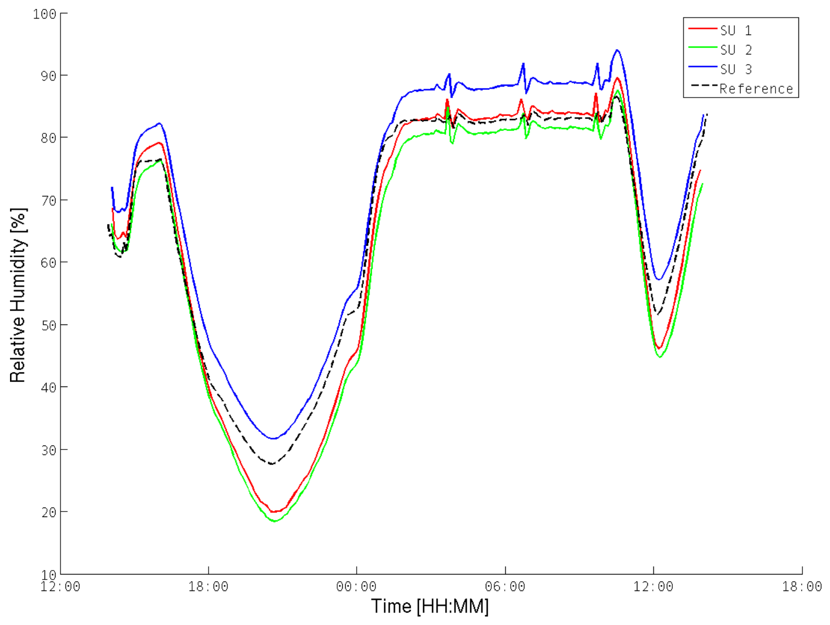


Figure 6.2: Humidity measurements in climate chamber, series1. The patterns are similar, but there are discrepancies up to 16% depending on temperature which is outside the sensors specification.

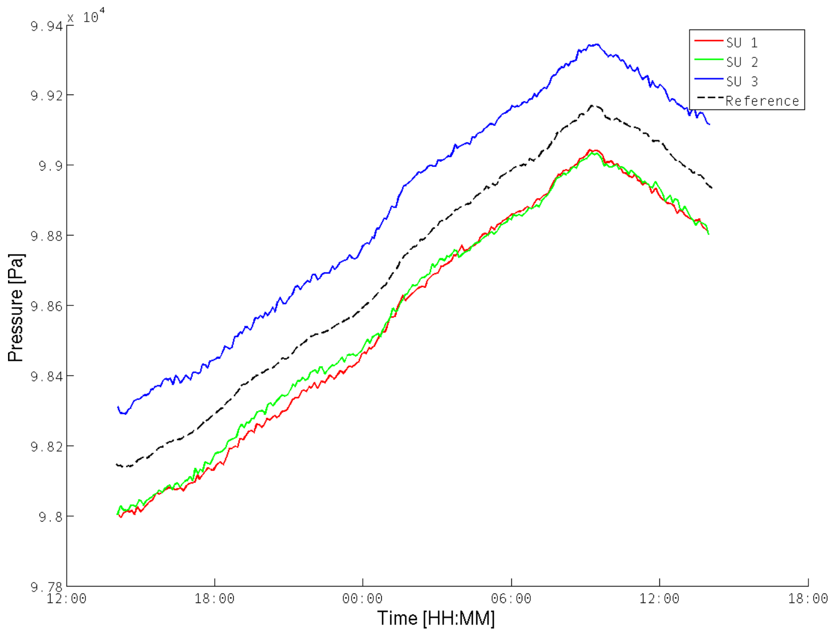


Figure 6.3: Pressure measurements in climate chamber, series1. Only 0.3% difference between the sensors with a constant offset.

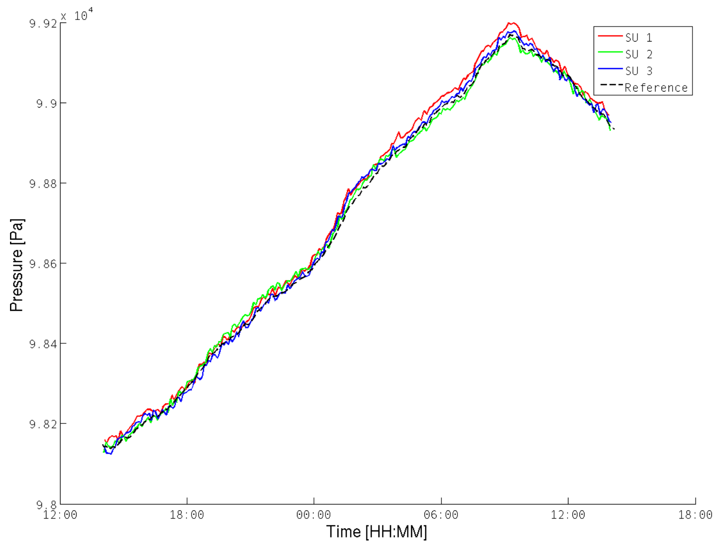


Figure 6.4: Pressure measurements with offset in climate chamber, series1. The results show that the offset is constant in this range.

6.1.2 Series2

Figure 6.5 describes temperature measurements in the climate chamber during constant temperature. The results in this section are the mean of every five measurements, taken once a minute. The results from the humidity measurements are omitted as they add no new information.

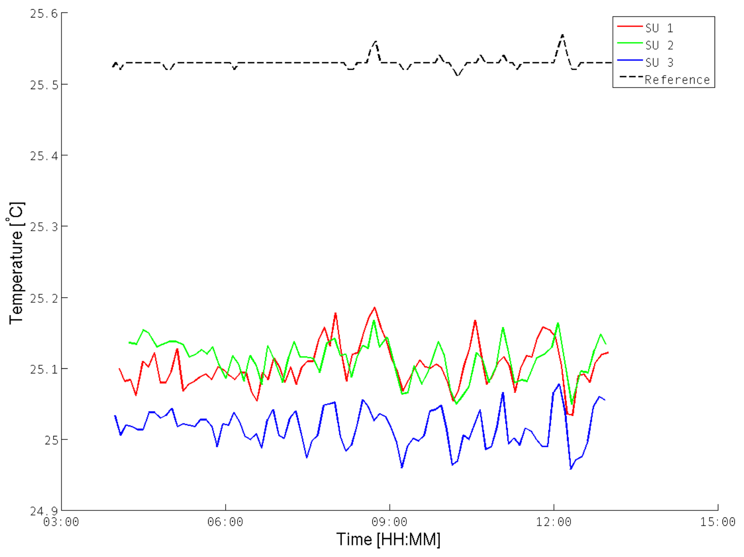


Figure 6.5: Measurements in climate chamber with constant temperature, series2. 0.5°C difference between SUs and reference.

6.2 Real world performance

6.2.1 Walk1

This section presents temperature, humidity, and pressure results from walk1 in Lund, see figure 6.6 for the route. During these tests the measurement setup was placed in a bicycle basket to reduce the impact of body heat. The SUs measurement interval is 20 seconds and the reference interval is 30 seconds without averaging.

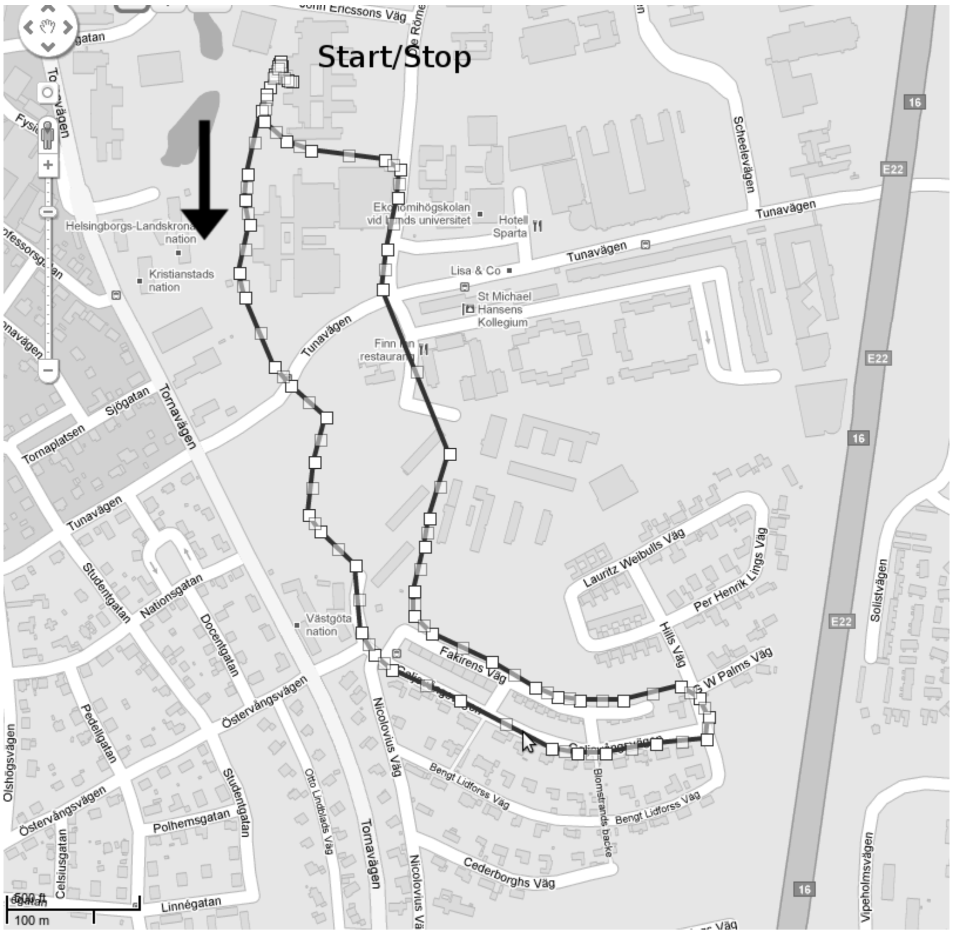


Figure 6.6: GPS data overlayed on Google Maps from walk1. Arrow indicates walking direction. Squares represent measurement points. ©2010 Google - Map data ©2010 Tele Atlas

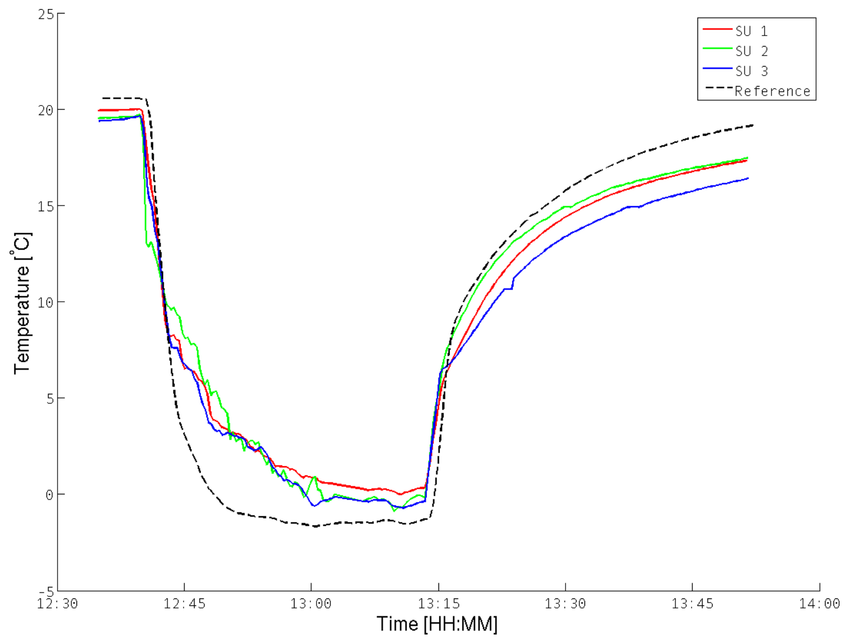


Figure 6.7: Temperature measurements during walk1. Note the difference in fall time between the reference and SUs.

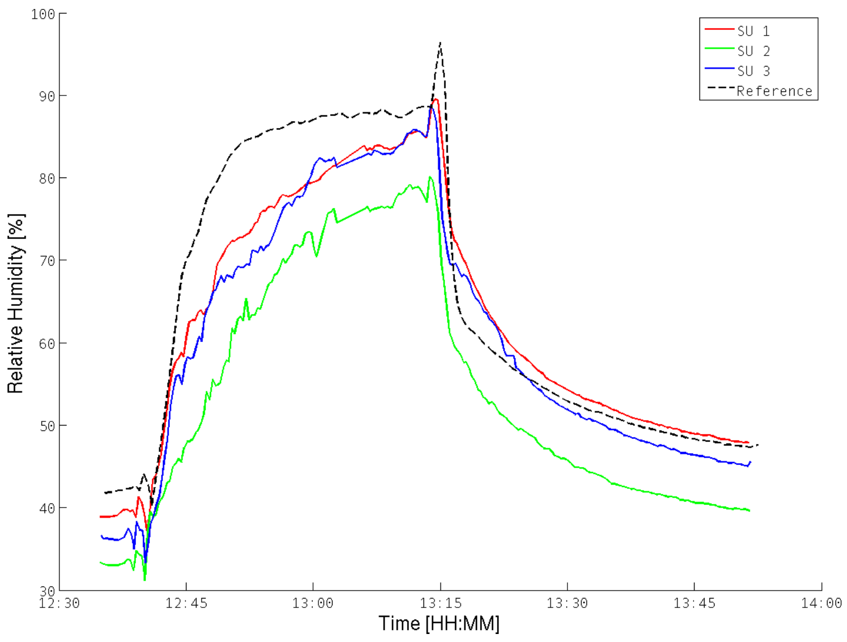


Figure 6.8: Humidity measurements during walk1. Note that the difference in temperature has a large effect on the humidity results. There is also quite a large spread internally between the SUs.

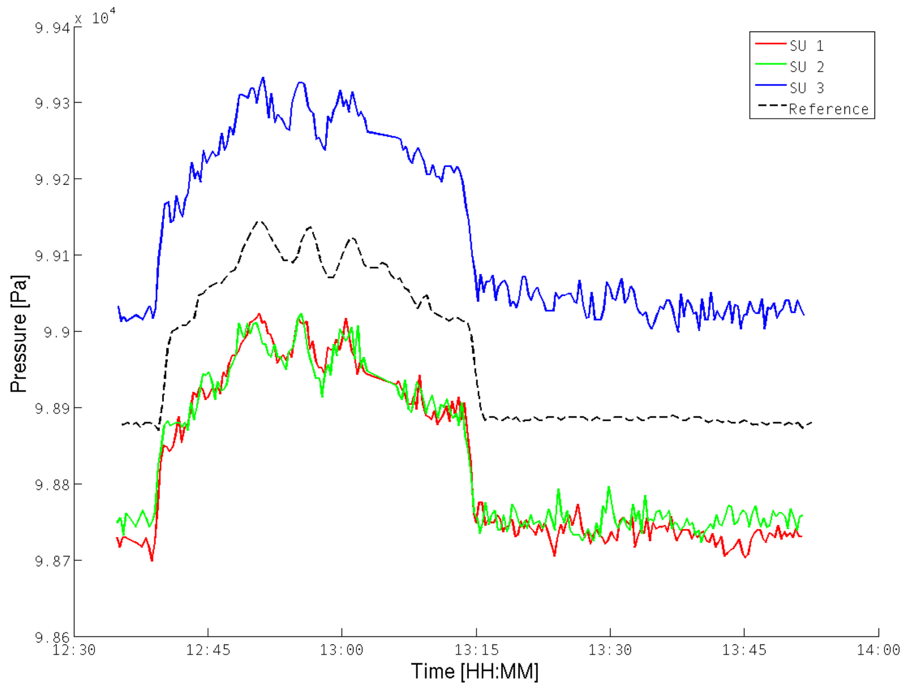


Figure 6.9: Pressure measurements during walk1. Constant offset present here as well.

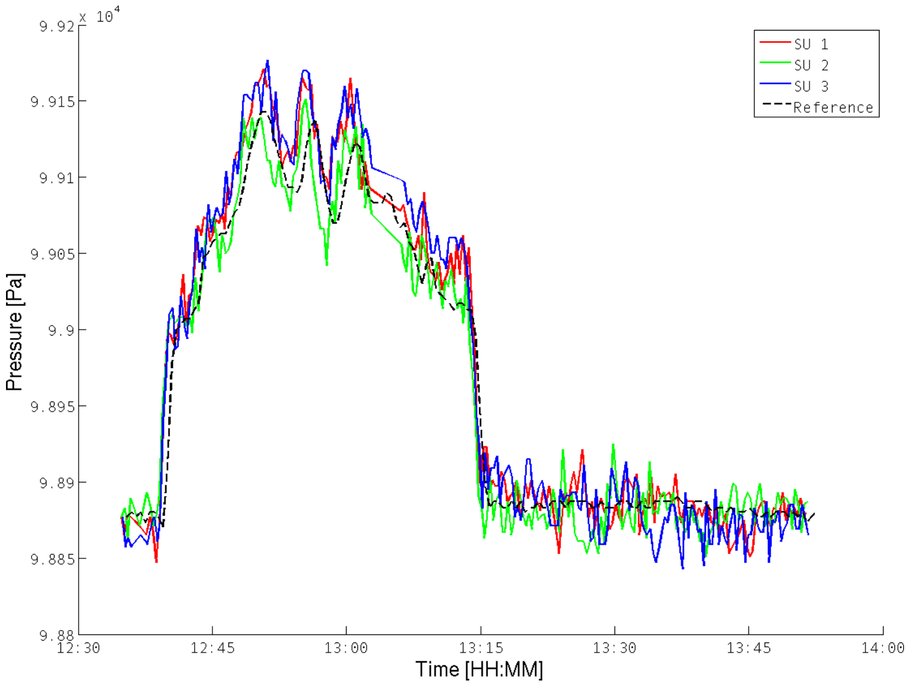


Figure 6.10: Pressure measurements with fixed offset during walk1.

Figure 6.11 shows the height changes derived from the pressure measurements during walk1 using equation 2.5.

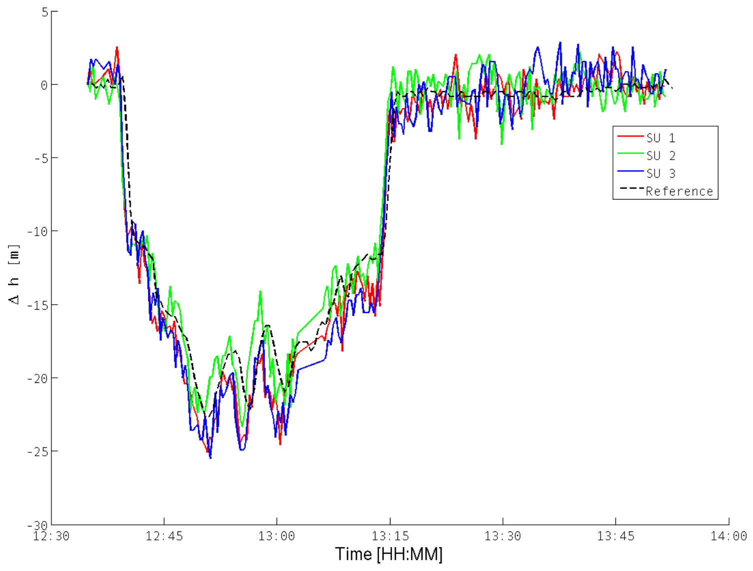


Figure 6.11: Δ h during walk1.

Figure 6.12 shows the altitude difference during walk1 using data from Google Earth, excluding data from indoor measurements. The results are quite similar compared to figure 6.11.

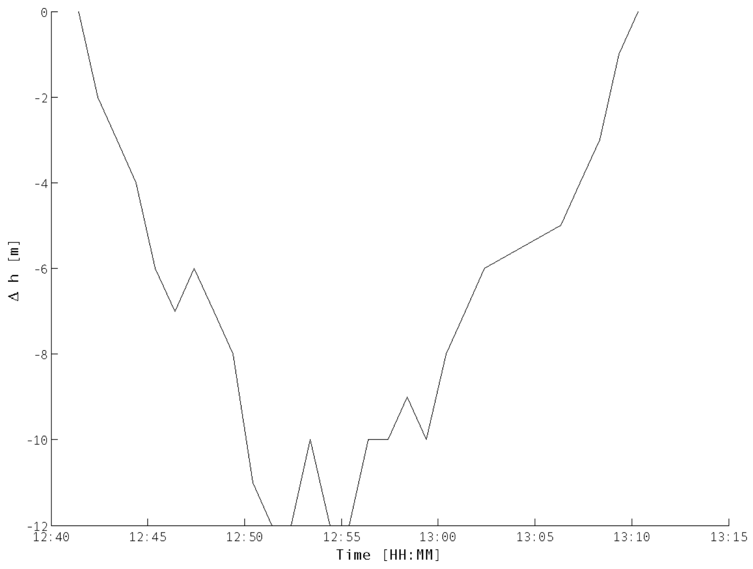


Figure 6.12: Δ h during walk1 extracted from Google Earth, indoor data not included. Altitude resolution: 1m. ©2010 Tele Atlas, ©2010 PPWK, ©2010 Geocentre Consulting.

6.2.2 Walk2

This section presents temperature and humidity results from walk2 in Lund, see figure 6.13 for the route. During these measurements the measurement setup was carried as a backpack, placing SU 2 closer to the users body. The SUs measurement interval is 20 seconds and the reference interval is 30 seconds without averaging.

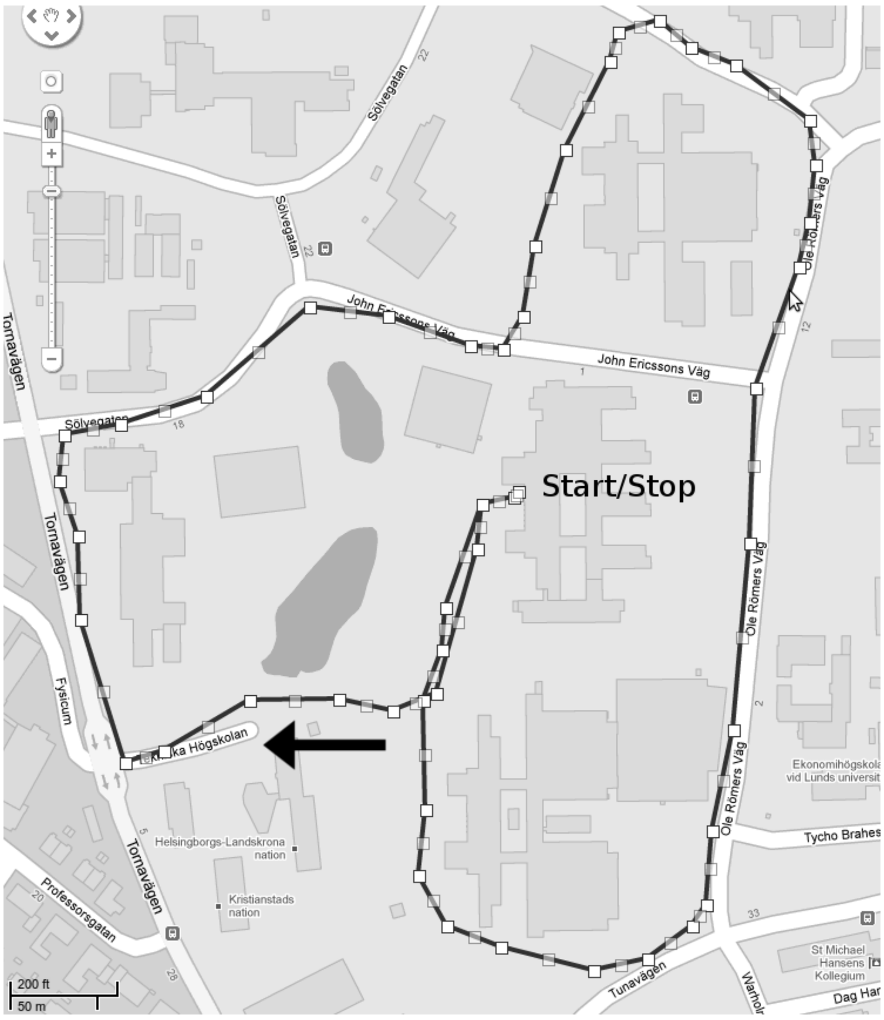


Figure 6.13: GPS data overlayed on Google Maps from walk2. Arrow indicates walking direction. Squares represent measurement points. ©2010 Google - Map data ©2010 Tele Atlas

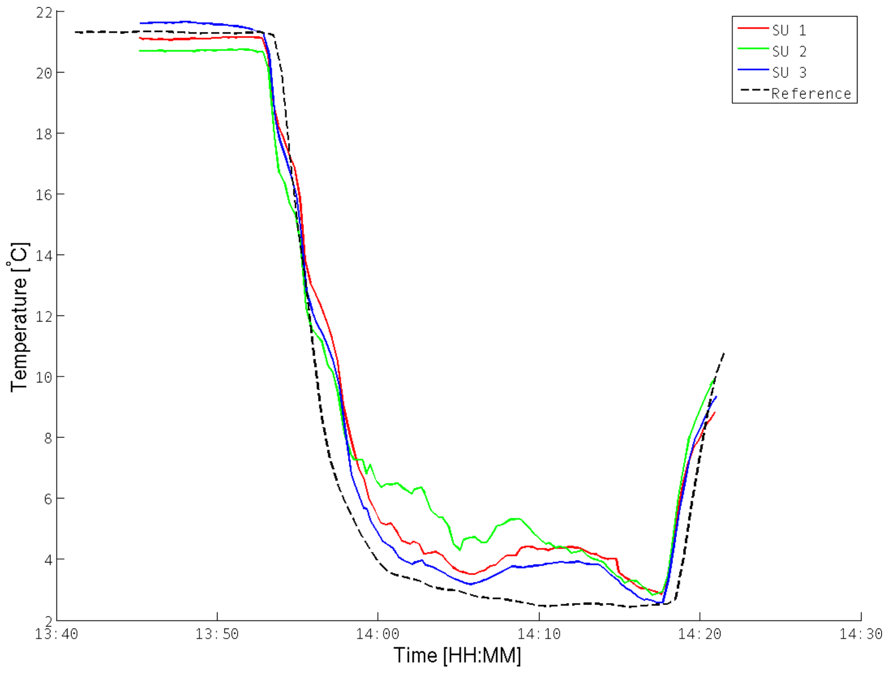


Figure 6.14: Temperature measurements during walk2. Note the higher temperature for SU 2 between 14:00 to 14:10.

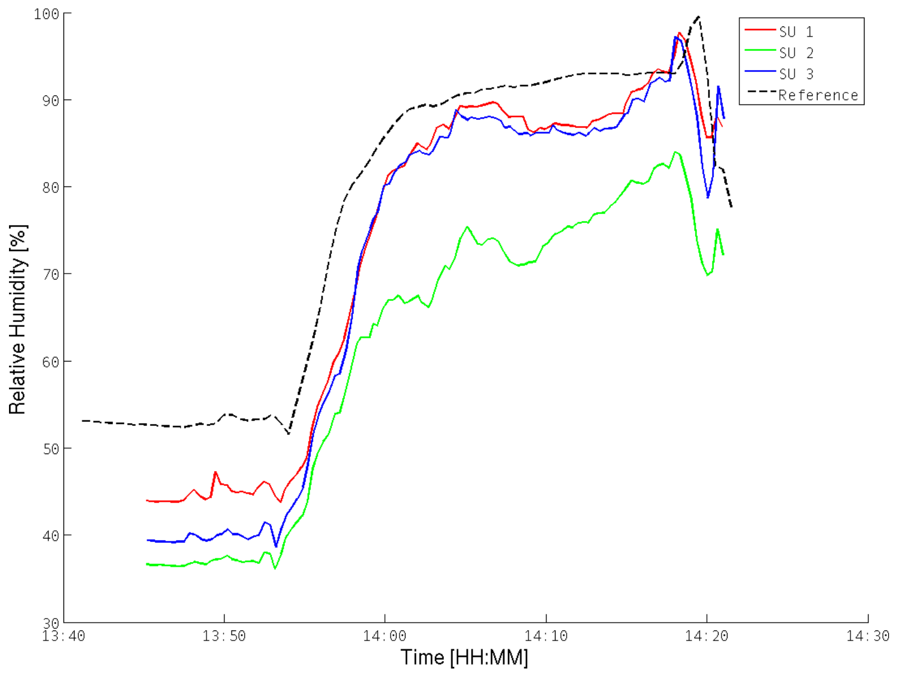


Figure 6.15: Humidity measurements during walk2. Note that SU 2 shows a slightly different behaviour than the other sensors.

6.2.3 Indoors

This section presents temperature and humidity results from a stationary, indoor measurement. The results in this section are the mean of every five measurements, taken once a minute.

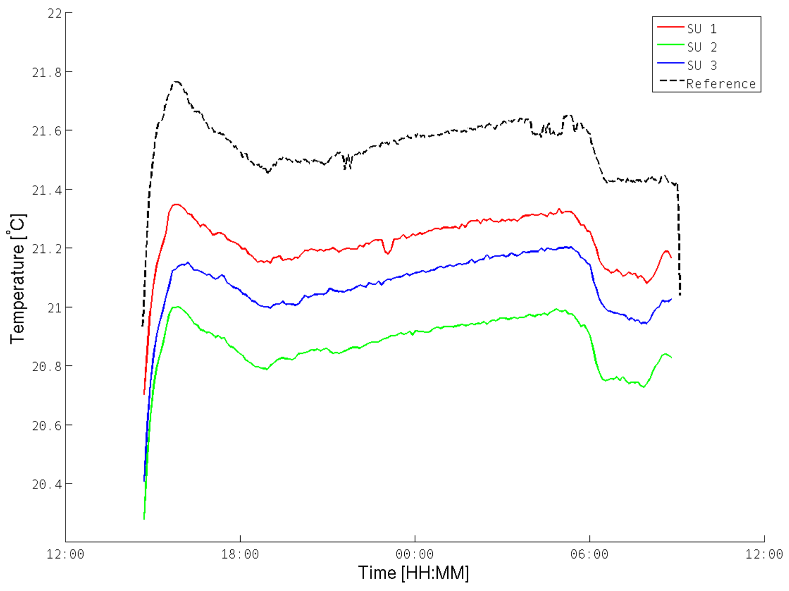


Figure 6.16: Temperature readings indoors. 0.4°C largest spread between SUs. 0.8°C largest spread between SUs and reference.

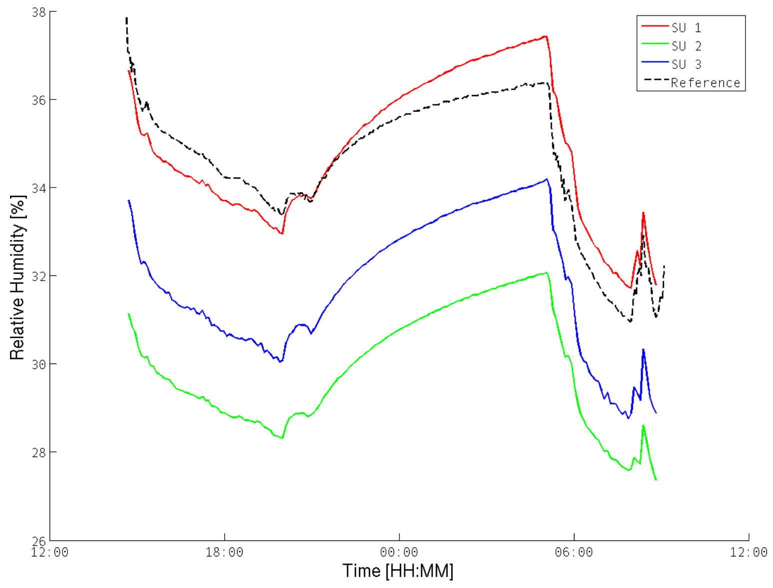


Figure 6.17: Humidity measurements indoors. Roughly 6% largest spread between SUs and reference.

6.3 Power consumption

The voltage drop measured over the resistor can be seen in figures 6.18 and 6.19. When idle the circuit draws 4.8mA, with a current increase to 20mA every 50ms, lasting for 2.5ms.

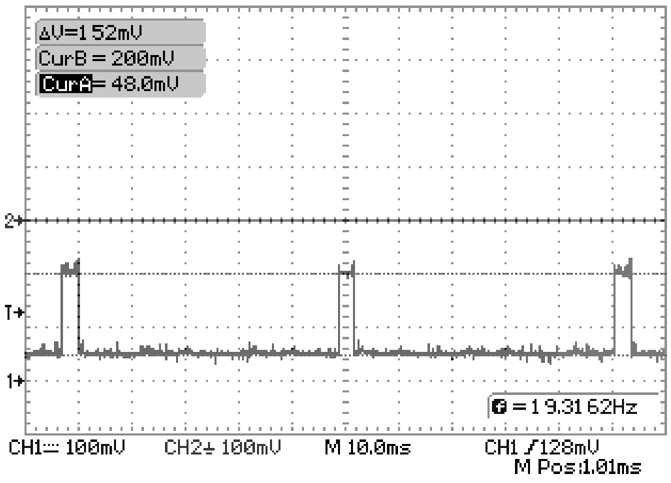


Figure 6.18: Voltage drop over the 10Ω resistor, Bluetooth idle.

With the Bluetooth module connected, the current drawn ranges from 4.4mA to 42mA, with an average of ~20mA.

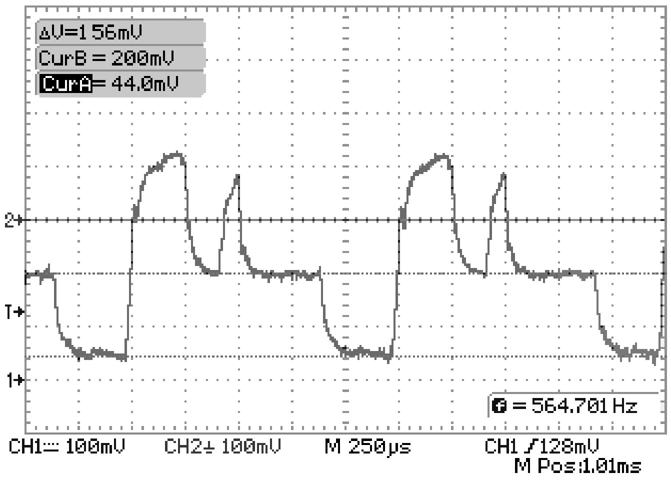


Figure 6.19: Voltage drop over the 10Ω resistor, Bluetooth connected.

During a battery test where a measurement was done once a minute, fully charged batteries lasted for 11 days.

Chapter 7
Discussion

7.1 Sensor unit design

The prototype sensor unit design is dependent on air flow for relevant temperature and humidity measurements. This explains why the rise time is slower than the fall time in figure 6.7, as the SUs were not moving when indoors. This is an issue, as most measurements indoors will be quite stationary compared to measurements outdoors.

The original idea proposed the sensors to be integrated in cell phones. Advantages would be easier operation for the user, no need for external communication as USB/Bluetooth, and better battery performance. Also, the need of the AVR would be omitted. Disadvantages include difficulties regarding temperature/humidity measurements and the issue of upgrading sensor hardware. The problems with temperature measurements can be seen in figure 6.14, where SU 2 does not entirely follow the pattern of SU 1 and SU 3. This is because SU 2 is mounted closer to the user and is therefore affected by body heat. Therefore, placing the temperature/humidity sensor in a cell phone is not a viable option as the user and phone will prevent the sensor from reaching ambient conditions. However, a pressure sensor could be integrated in a cell phone as long as it is temperature compensated and not placed in a sealed compartment.

When investigating the different solutions for communication between SU and phone, USB was one of the options. However, due to the current lack of cell phones supporting USB host mode, Bluetooth was chosen for the prototype.

7.1.1 Component cost

The component cost of the SU can be seen in table 7.1.

Component	**Cost [SEK.]**
AVR ATtiny45	26
BMP085	149
SHT1x	289
BlueSMiRF Gold	459
Box	41
Battery pack	21
Batteries	104
Voltage regulator	20
Total	**1109**

Table 7.1: Sensor unit cost

The most expensive part is the Bluetooth module. If the SU was redesigned to connect by USB instead, the cost would be cut almost in half. Removing the Bluetooth module, the size of the PCB could also be decreased, lowering production costs. Another possibility of decreasing costs but keeping Bluetooth communication is to implement the Bluetooth stack on an AVR in combination with a radio chip.

7.1.2 Power consumption and power supply

To decrease the power consumption, the page and inquiry scan window on the Bluetooth module were decreased to the smallest possible value, corresponding to a duty cycle of about 0.9%. Furthermore, the transmit power of the Bluetooth module can be decreased. The maximum transmit power is 12 dBm, and the minimum -20 dBm. Decreasing the window and transmit power parameters will have a positive effect on power consumption, but will in turn make it harder to connect to the Bluetooth module.

Rechargeable NiMH batteries were chosen as power supply as they provide a cost effective alternative to the more expensive lithium-ion batteries. However, lithium-ion batteries would provide a smaller design as they can provide a higher energy density [29]. Furthermore, if communication between the application and SU was changed to USB, both batteries and the Bluetooth module could be removed completely.

7.2 Measurements

Factors as newly charged batteries and high measurement frequency had a negative impact on the quality of the measurements. This as batteries became warm during charging and the internal logic of the sensors generated heat during excessive usage. According to the data sheet of the SHTxx [9][10], the sensor should not be active for more than 10% of the time to avoid self heating.

Following factors are important regarding the performance of the sensors:

- The radiation shield. A correctly designed radiation shield should minimize radiation but still allow for proper ventilation.
- The color spectrum of the radiation shield should be white for all types of radiation.
- Materials used around the sensors should not have high heat inertia and should not absorb moisture [9].

7.3 Application

The SU is compatible with different software platforms. It has been tested with Android 2.1/2.2 and Ubuntu 10.04 using Bluez [28].

The Bluetooth implemenation in HTC's Legend cell phone was discovered to be problematic. While comparing against other HTC based Android phones, it became clear that the HTC Legend sometimes had problems connecting to the SU, while the others worked without issues. Tests were also made with a custom version of Android 2.1[1] and 2.2[2] on a HTC Hero. Both these versions of Android also worked without issues. To solve the connection problems that HTC Legend has, multiple connection attempts are made from the application when connecting to the SU. Therefore, connecting to the SU from HTC Legend sometimes takes up to 2 minutes.

As the system is designed currently, all user collected data is sent to the server one-way. To make the system appealing for the general population, the application needs functionality to receive data from the system according to the users wishes. Information such as local weather forecasts and traffic predictions could be interesting for the general user.

[1]http://forum.xda-developers.com/showthread.php?t=638584
[2]http://forum.xda-developers.com/showthread.php?t=784689

7.4 Sensor Performance

Depending on the type of measurement, the performance of the sensors varied. Temperature measurements came close to the performance of the reference, see figure 6.5. The mean values for SU 1, SU 2, SU 3, and reference can be seen in table 7.2. The SUs spread is within the manufacturers specification, the difference between the SUs and the reference could be due to sensor drift of the refence. Verification of the reference itself would be needed. Figure 6.1 shows that the accuracy and precision is still applicable in a larger temperature range.

Sensor unit	**Mean Temperature [°C]**
SU 1	25.16
SU 2	25.18
SU 3	25.07
Reference	25.76

Table 7.2: Mean values of the temperature measurements from figure 6.5.

Table 7.3 presents the mean offsets compared to the reference during the pressure measurements in figures 6.3 (series1) and 6.9 (walk1).

Sensor unit	**Pressure offset, figure 6.3 [Pa]**	**Pressure offset, figure 6.9 [Pa]**
SU 1	-137.96	-140.65
SU 2	-118.12	-131.85
SU 3	175.67	163.18

Table 7.3: Mean pressure offsets compared to the reference.

As the mean offsets of the pressure sensors are relatively similar, the error can be reduced by introducing an offset. This is illustrated in figures 6.10 and 6.4. To verify that the offsets are present for the entire range of the sensors, specific pressure testing should be done in a pressure chamber. This was not possible due to lack of equipment.

In figure 6.11, the first pressure measurement for each sensor is set to p_0 in equation 2.5. This will give all sensors the same starting point at Δ h 0m. The natural pressure variation will affect the altitude measurements, but as walk1 only lasted for approximately one hour the effect will be small. According to the pressure measurements, the difference between the lowest[3] and highest[4] point outdoors is 13m. The same points in Google Earth show a difference of 12m. This shows that the pressure sensor is sensitive enough to register altitude variations that occur in every day situations.

The results from the humidity measurements show the same trend as the reference. Because of the temperature inertia, the results from the humidity measurements during large temperature variations will lag compared to the reference,

[3]Figure 6.11 and 6.12 at 12:41

[4]Figure 6.11 and 6.12 at 12:55

see figure 6.8. However, the temperature variations do not entirely explain the results in figure 6.2. According to the temperature readings in figure 6.1, where the reference displays the highest temperature at 21:00, it should show the lowest RH at the same time in figure 6.2. Instead, SU 1 and SU 2 displays the lowest RH, indicating there are other factors than the temperature inertia affecting the humidity measurements. One of these factors is probably because of the data from the CS215 probe being corrected in the logger software. That is, the CS215 probe has been calibrated both from factory and at a later instance, while the temperature/humidity sensors in the SUs are only calibrated from factory. Therefore, the SUs should include a software calibration to improve performance. Figure 6.8 shows that the humidity offset is not constant for the entire humidity range, independant of sensor version and package. This needs to be taken into account during calibration. As the three temperature/humidity sensors used in the prototypes are two different versions, indicating different production dates, software recalibration is even more important as the sensors have a long term drift of <0.5%RH/year [9].

After contacting Sensirion to verify the humidity results it was concluded that a probable cause to the spread was due to chemical out-gasing either from the PCB or the box itself. The gases could have permanently damaged the capacitive sensing element as it is sensitive to acids and solvents [30].

7.5 Future improvements

To improve the inertia of the SU, the SU box needs to be redesigned. The temperature/humidity sensor should be placed as freely as possible with an appropriate radiation/weather shield. In figure 7.1, a possible design is presented. The AVR and pressure sensor are placed in a cylinder, with the temperature/humidity sensor extruding from the bottom, surrounded by the shielding.

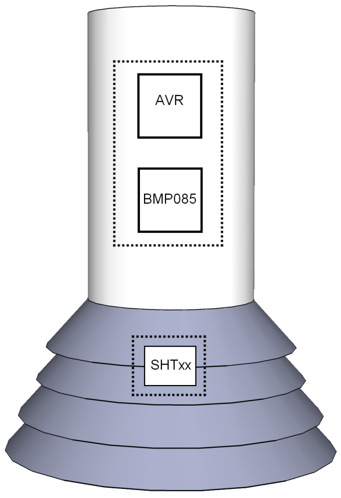


Figure 7.1: Improved sensor unit design.

The cylinder is completely encapsulated except at the bottom, where an opening allows for correct pressure measurements. The top of the cylinder consists of a universal plug, which can connect to either a USB port or directly to a combined Bluetooth/battery pack. A possible extension could include a GSM modem/battery pack, making the SU an independent module. This opens for the possibility of three different versions of the SU; independent GSM/battery prepared, USB prepared, and Bluetooth prepared SUs. The highly mobile USB SUs would be carried by citizens, while the independent SUs could be placed at remote locations, providing data not represented by individuals. The Bluetooth prepared SUs would be provided to citizens as an alternative to the USB version, providing higher mobility at the cost of a higher price.

The temperature/humidity sensor used in this design is available in a corresponding version with an I^2C interface. This would decrease the number of sensor protocols used on SU from two to one and reduce the number of pins used on the AVR by one. Since the I^2C bus allows for multiple slaves, the SU can be modularized, meaning sensors could be added and removed freely. To implement this functionality, the AVR firmware and application would need to be updated each time a new sensor is added. To simplify this process, the AVR could be programmed with a general function that forwards I^2C commands from the cell phone application. This would mean that when adding a new sensor, only the application side would be affected.

To reduce the size of the PCB, components could be surface mounted. This in turn would make it possible to decrease the size of the entire product.

The power consumption could be reduced by letting the AVR control the power supply for the sensors. Also, by disabling parts of the AVR not used, even more battery time can be saved. Another way of reducing the power consumption is to make the SU connect to the phone instead. The AVR could keep the Bluetooth module completely shut down and only enable it when data needs to be sent.

The Android application could be improved with precise positioning in places where the GPS signal is weak or unavailable. This could be accomplished by using the internal accelerometers in combination with the GPS and compass. Furthermore, a service could be added alongside the application, handling the time scheduling instead of the TimerTask in the main thread. This would ensure more precise time intervals between each measurement, as the interval specified in the TimerTask may drift somewhat depending on the amount of resources available on the phone.

Chapter 8

Conclusion

The hardware and software prototype presented in this thesis show that it is possible to design a mobile sensor unit to be carried by users. The collected data could be used to complement already existing global data sets [2]. Furthermore, the sensors do not need to be limited to persons but can be placed in or on other objects e.g. vehicles, creating possibilities for other applications.

However, design decisions regarding sensors, encapsulation, power consumption, cost, and application design have to be made. Also, tests have to be performed against a large user base to see how individuals will actually use the technology. Some valid questions are:

- Do people want to participate in these kinds of measurements?
- How many users are required per square kilometer for good coverage?
- Will users intentionally influence measurements?
- What user feedback is needed to make the measurements interesting for the general population?

References

[1] Sheila Kinkade, Katrin Verclas, *Environmental Monitoring with Mobile Phones*, Wireless Technology for Social Change, UK: UN Foundation - Vodafone Group Foundation Partnership, 2008

[2] Peter A. Stott and Peter W. Thorne, *How best to log local temperatures?*, nature, Vol 465, 13 May 2010

[3] K Birkelund, Kim Degn Jensen, Emil Højlund-Nielsen, Johan Nagstrup, Anders Lei, Søren Dahl Petersen, Andrea U Andreassen and Erik V Thomsen, *MEMS Climate sensor for crops in greenhouses*, Journal of Micromechanics and Microengineering, 20 (2010) 085021

[4] Niwat Thepvilojanapong, Takahiro Ono and Yoshito Tobe, *A Deployment of Fine-Grained Sensor Network and Empirical Analysis of Urban Temperature*, Sensors, ISSN 1424-8220

[5] Matthew Williams, Dan Cornford, Lucy Bastin, Richard Jones, Stephen Parker, *Automatic processing, quality assurance and serving of real-time weather data*, Computers Geosciences, doi:10.1016/j.cageo.2010.05.010

[6] Lama Nachman, Jonathan Huang, Raymond Kong, Rahul Shah, Junaith Shahabdeen, Chieh-yih Wan, Mark Yarvis, *On-body health data aggregation using mobile phones*, Corporate Technology Group, Intel Corp.

[7] Philips Semiconductors, *THE I^2C-BUS SPECIFICATION, VERSION 2.1*, `http://www.nxp.com/acrobat_download2/literature/9398/39340011.pdf`

[8] Bosch Sensortec, *BMP085 Digital Pressure Sensor, Data Sheet*, `http://www.bosch-sensortec.com/content/language1/downloads/BST-BMP085-DS000-05.pdf`

[9] SENSIRION, *Datasheet SHT1x (SHT10, SHT11, SHT15)*, `http://www.sensirion.com/en/pdf/product_information/Datasheet-humidity-sensor-SHT1x.pdf`

[10] SENSIRION, *Datasheet SHT7x (SHT71, SHT75)*, `http://www.sensirion.com/en/pdf/product_information/Datasheet-humidity-sensor-SHT7x.pdf`

[11] SENSIRION, *Introduction to Humidity, Basic principles on Physics of Water Vapor*,
`http://www.sensirion.com/en/pdf/product_information/Introduction_to_Relative_Humidity_E.pdf`

[12] Frank Durda, *Serial and UART Tutorial*,
`http://www.freebsd.org/doc/en/articles/serial-uart/`

[13] Bluetooth SIG, *How it works*,
`http://www.bluetooth.com/English/Technology/Works/Pages/default.aspx`

[14] Android, *What is Android?*,
`http://developer.android.com/guide/basics/what-is-android.html`

[15] Paul A. Tipler, Gene Mosca, *PHYSICS For Scientists and Engineers, 6th ed. p. 577*,
W.H Freeman and Company, 2008

[16] Paul A. Tipler, Gene Mosca, *PHYSICS For Scientists and Engineers, 6th ed. p. 575, equation 17-16*,
W.H Freeman and Company, 2008

[17] Atmel, *8-bit AVR Microcontroller with 2/4/8K Bytes In-System Programmable Flash*,
`http://www.atmel.com/dyn/resources/prod_documents/doc2586.pdf`

[18] National Semiconductor, *LP3852/LP3855 1.5A Fast Response Ultra Low Dropout Linear Regulators*,
`http://www.national.com/profile/snip.cgi/openDS=LP3855`

[19] Parallax inc., *Sensirion temperature/humidity sensor*,
`http://www.parallax.com/StoreSearchResults/tabid/768/txtSearch/28018/List/0/SortField/4/ProductID/94/Default.aspx`

[20] Roving Networks, *RN-41 Class 1 Bluetooth Module*,
`http://www.rovingnetworks.com/documents/RN-41.pdf`

[21] Spark Fun Electronics, *BlueSMiRF RN-v1*,
`http://www.sparkfun.com/datasheets/RF/BlueSMiRF-Gold-ChipAnt-v1.pdf`

[22] Apple, *iOS Dev Center*,
`http://developer.apple.com/devcenter/ios/index.action`

[23] Nokia, *maemo.org*,
`http://maemo.org/intro/`

[24] Nokia s60 devices, *Device specifications s60*,
`http://www.forum.nokia.com/Devices/Device_specifications/?filter=s60`

[25] Nokia, *SYMBIAN*[1],
http://www.symbian.org/

[26] Roving Networks, *Roving Networks Bluetooth Product User Manual*,
http://www.sparkfun.com/datasheets/Wireless/Bluetooth/rn-bluetooth-um.pdf

[27] Campbell Scientific, Inc.,
CS215 Temperature and Relative Humidity Probe,
http://www.campbellsci.com/documents/manuals/cs215.pdf

[28] BlueZ, *BlueZ, Official Linux Bluetooth protocol stack*,
http://www.bluez.org/

[29] Chester Simpson, *Characteristics of Rechargeable Batteries*,
http://www.national.com/appinfo/power/files/f19.pdf

[30] SENSIRION, *Handling Instructions For SHTxx Humidity and Temperature Sensors*,
http://www.sensirion.com/en/pdf/product_information/Handling_Instructions_V1.1_C1.pdf

[1]Website shutting down 17 Dec. 2010.

Appendix A

SHTxx data conversion

Equations A.2, A.3 and A.1 are taken from the SHTxx data sheet, see reference [9] and [10].

Temperature calculation:

$$T = d_1 + d_2 \cdot SO_T \tag{A.1}$$

where

$$d_1 = \textit{-39.6}$$
$$d_2 = \textit{0.01}$$
$$SO_T = \textit{Sensor output, temperature}$$

Relative Humidity calculation:

$$RH_{linear} = c_1 + c_2 \cdot SO_{RH} + c_3 \cdot SO_{RH}^2 \tag{A.2}$$

where

$$SO_{RH} = \textit{Sensor output, humidity}$$

Constant	**Version 4**	**Version 3**
c_1	-2.0468	-4.0
c_2	0.0367	0.0405
c_3	$-1.5955 \cdot 10^{-6}$	$-2.8 \cdot 10^{-6}$

Table A.1: Version 3 and version 4 constants.

$$RH_{true} = (T - 25) \cdot (t_1 + t_1 \cdot SO_{RH}) + RH_{linear} \tag{A.3}$$

where

$$t_1 = \textit{0.01}$$
$$t_2 = \textit{0.00008}$$

Appendix B

BMP085 data conversion

Equations B.1 and B.2 are taken from the BMP085 data sheet, see reference [8].

Temperature calculation:

$$\begin{aligned} X1 &= (UT - AC6) \cdot (AC5/2^{15}) \\ X2 &= MC \cdot 2^{11}/(X1 + MD) \\ B5 &= X1 + X2 \\ T &= (B5 + 8)/2^4 \end{aligned} \tag{B.1}$$

Pressure equation:

$$\begin{aligned} B6 &= B5 - 4000 \\ X1 &= (B2 \cdot (B6 \cdot B6/2^{12}))/2^{11} \\ X2 &= AC2 \cdot B6/2^{11} \\ X3 &= X1 + X2 \\ B3 &= ((AC1 \cdot 4 + X3) \cdot 2^{oss} + 2)/4 \\ X1 &= AC3 \cdot B6/2^{13} \\ X2 &= (B1 \cdot (B6 \cdot B6/2^{12}))/2^{16} \\ X3 &= ((X1 + X2) + 2)/4 \\ B4 &= AC4 \cdot (X3 + 32768)/2^15 \\ B7 &= (UP - B3) \cdot (50000/2^{oss}) \\ P &= (B7 \cdot 2)/B4 \\ X1 &= (P/2^8) \cdot (P/2^8) \\ X1 &= (X1 \cdot 3038)/2^{16} \\ X2 &= (-7357 \cdot P)/2^{16} \\ P &= P + (X1 + X2 + 3791)/2^4 \end{aligned} \tag{B.2}$$

AC1-AC6, B1, B2, MB-MD are calibration constants from the EEPROM. UT and UP are the sensor outputs for temperature and pressure respectively.

Appendix C
XML-format

Depending on the internal sensors available, the size of the comment field, and which positioning system is used, related XML tags will be added or removed.

```
<data>
<source>3</source>
<phoneModel>HTC Hero</phoneModel>
<sensor_mac>00:06:66:05:04:08</sensor_mac>
<day>2010-11-26</day>
<time>15:03:27</time>
<su_version>69</su_version>
<sht_model>11</sht_model>
<sht_version>3</sht_version>
<position_latitude>55.697351</posistion_latitude>
<position_longitude>13.215379</posistion_longitude>
<position_accuracy>2370.0</position_accuracy>
<temperature>21.46</temperature>
<humidity>29.89172</humidity>
<bmptemp>21.4</bmptemp>
<pressure>99610</pressure>
<lightSensor>Capella Microsystems CM3602 Light sensor 1</lightSensor>
<light>225.0</light>
<accelSensor>The Android Open Source Project BMA150 3-axis Accelerometer 1</accelSensor>
<accelerationx>-1.1441092</accelerationx>
<accelerationy>6.932757</accelerationy>
<accelerationz>6.0474343</accelerationz>
</data>
```